

**Bachelor Thesis**

-

**Analysis and simplification of architectural floor plans**

Principals:	PlanFabrik GmbH
Authors:	Alexander Wyss, Florian Bruggisser
Supervising Prof.:	Simon Schubiger, Stefan Arisona
University:	FHNW Technik
Email:	alexander.wyss@students.fhnw.ch florian.bruggisser@students.fhnw.ch

January 22, 2017

## **1 Abstract**

This work's main topic is the analysis of floor plans to detect and mark areas where different types of floor heatings will be installed. The idea is to automate this process for simple floor plans and provide additional tools to correct possible errors in more complicated plans.

This work will be split into several parts. The first part will describe several algorithms used for preprocessing the image to remove any noise. Additionally, the algorithms for room detection and object detection will be explained.

The second part will contain the whole project management and implementation. This includes for example a timetable of our work, the class structure and much more information concerning the project setup and execution.

Last but not least, we will present and discuss the results of our work. This also includes features that may be built in the future and the limitations of our work.

## Contents

1	Abstract . . . . .	1
2	Introduction . . . . .	3
	2.1 Starting position . . . . .	3
	2.2 Problem description . . . . .	3
	2.3 Goals . . . . .	3
3	Theory . . . . .	4
	3.1 Information Segmentation . . . . .	4
	3.2 Structural Analysis . . . . .	4
	3.3 Semantic Analysis . . . . .	4
4	Implementation . . . . .	6
	4.1 Experiment . . . . .	6
	4.2 Parameter Finding . . . . .	6
	4.3 Information Segmentation . . . . .	6
	4.4 Structural Analysis . . . . .	6
	4.5 Semantic Analysis . . . . .	6
5	Software Architecture . . . . .	7
	5.1 Input / Output . . . . .	8
	5.2 Algorithm . . . . .	8
	5.3 Workflow . . . . .	9
	5.4 User Interface . . . . .	9
6	Result . . . . .	10
7	Conclusion . . . . .	11
8	Developer Guide . . . . .	12
	8.1 Add Algorithm . . . . .	12
	8.2 Edit UI . . . . .	12

## **2 Introduction**

### **2.1 Starting position**

The company PlanFabrik GmbH creates floor plans for house technology. In a first step the plans are analysed and enhanced through an employee. He adds more information to the plan, like room polygons or he marks areas where house technology can be installed and where not. Those areas are defined by different features. For example, usually the area close to windows has a higher density of pipes emitting heat to the room. This is to equal out the heat loss that usually occurs on windows.

The process of drawing those areas is currently tool supported but still takes a lot of time, equal to the rising amount of rooms the employee has to analyse. The idea is to create a more automated system which does require only a small amount of user input.

### **2.2 Problem description**

To manually create all the room-areas, it takes the user a lot of time. This is due to the fact that he has to click every corner of each room to make the room-enclosing polygon. The fact that the current tool is not too precise, which means that errors in selecting the exact corner exist regularly. The user then has to rearrange the created corner to fit the exact corner in the image. This process of selecting all the corners by hand is highly inefficient, especially for floor plans that have several hundred or even thousands of corners in it.

An automated solution would provide a faster and more comfortable solution for the client. Additionally, the software may provide additional info that is needed. It can calculate the size of each room directly, which would otherwise have to be calculated by hand.

### **2.3 Goals**

The idea of theory of this bachelor thesis is to compare different algorithms which can automatically analyse floor plans and draw the polygons for the rooms. The goal in the end is to show and explain a way to solve the problem and decide which algorithms are used best. All of this will help us create a software that can do the automated analysis for basic floor plans. More complicated floor plans may hold problems that our algorithms can not find, those are supposed to be solved manually with an editor.

In the end, the program is supposed to create an output-file with the information of the areas found which is exportable to the existing software that currently handles the following processes.

### 3 Theory

#### 3.1 Information Segmentation

##### Erosion and Dilation

In our project we need erosion to strengthen small features that we want to be highlighted as well as to get rid of small details in the picture itself. We use it in early preprocessing. It helps together with the dilation to clean up pictures to further process them without too many interfering objects in the image. The main usage is to extract an image that features all the walls and has as few as possible other lines on it. [1]

For our project we use the binary grayscale erosion. The erosion uses the following principle to work.

$$A \ominus B = \{z \in E | B \subseteq A\}$$

$$\text{where : } B_z = \{b + z | b \in B\} \text{ with } \forall z \in E$$

A is a binary image in the Euclidean space or an integer grid. The erosion is done with the structuring element B on the image A. The structuring element has to be a subset or equal to A, otherwise there will be no erosion. The mask B will be applied to all pixels possible, if the mask fits on the center pixel of A, the value will be retained. Otherwise it will get deleted. This means that only when B is completely contained in A, values of pixels are retained.

Binary dilation uses the exact same process. The only difference compared to the erosion is that the deletion of the pixel will not be setting pixel values to black (0) but to white (1). Both of those processes are inherently the same and can be summed up under the term morphological operation. These are altering the image with a mask, as we did here for erosion and dilation.

##### Distance transformation

#### 3.2 Structural Analysis

##### Orientated FAST and Rotated BRIEF

##### Cascade training

#### 3.3 Semantic Analysis

##### Hough transformation

##### Watershed

The watershed-algorithm in our project is used for segmentation of the different rooms. It can find rooms indifferent of its shape.

Write  
why  
SURF  
not used:  
License

The algorithm is processed on a grayscale image on which the color intensity is analogous to the height in a height map. The watershed in use does flooding. The idea is to place a water source in each regional minimum and flood the entire relief. It will stop if it meets a different watersource or an impassable barrier.

## 4 Implementation

### 4.1 Experiment

### 4.2 Parameter Finding

### 4.3 Information Segmentation

### 4.4 Structural Analysis

### 4.5 Semantic Analysis

Description  
of Test-  
set

Parameter  
finding  
and de-  
scription

Noise  
Removal  
(Mor-  
phologi-  
cal)

Distance  
Trans-  
form,  
Why not  
Contour  
Dete-  
ction

Object  
Dete-  
ction:  
TM,  
Hough  
Lines,  
ORB

Cascading  
Clas-  
sifier:  
To few  
samples,  
Cat  
images  
prove,  
more  
positive,  
thick-  
ening,  
polydp,  
positive  
negative  
ratio,  
more  
specific  
negative  
samples

## 5 Software Architecture

This chapter describes the solution developed during our work. At the beginning the software architecture will be explained, followed by the partial steps and experiments which were needed to find the solution.

The software architecture has to fulfil multiple requirements for this project. It should be easily extendable with new algorithms and ui extensions. The input and output format of the application should be independent from the algorithms to support different file types like bitmap images or DXF/DWG formats. The algorithms of the application should be link together as workflows which then can be executed by an engine in parallel.

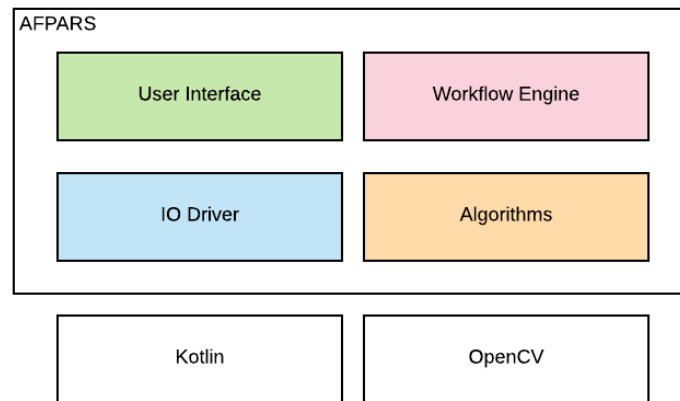


Figure 1: AFPARS software architecture.

To fulfil these requirements, the architecture of the software is split into four parts as shown in Figure 1. The complete application is based on the Java Virtual Machine (JVM) where Kotlin is running on. For image processing and recognition the application uses the library OpenCV.



## 5.1 Input / Output

To support the different input and output formats the architecture uses a meta format for the floor plan images called AFIImage. Different drivers add the support for multiple file formats. With this architecture it is possible to extend the software with new file formats and work internally with the meta container.

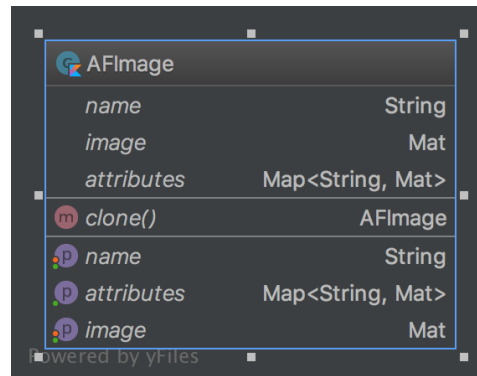


Figure 2: AFIImage class diagram.

The meta container AFIImage contains multiple attributes (see Figure 2) which can be used by algorithms to get information created by other algorithms or store information into an existing image.

## 5.2 Algorithm

blas

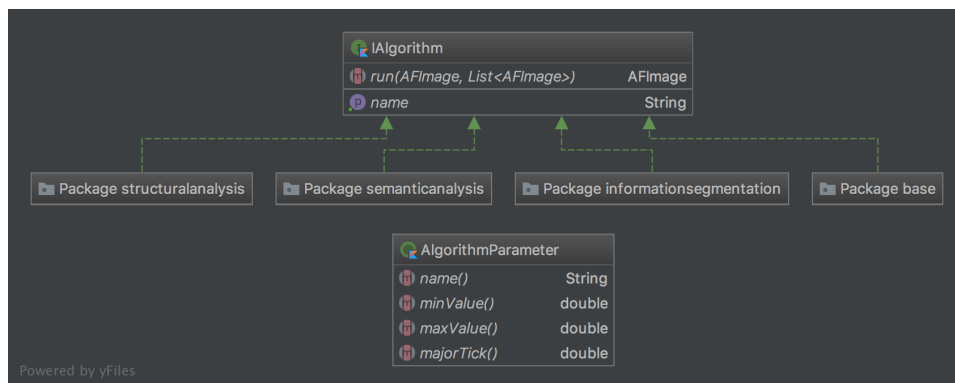
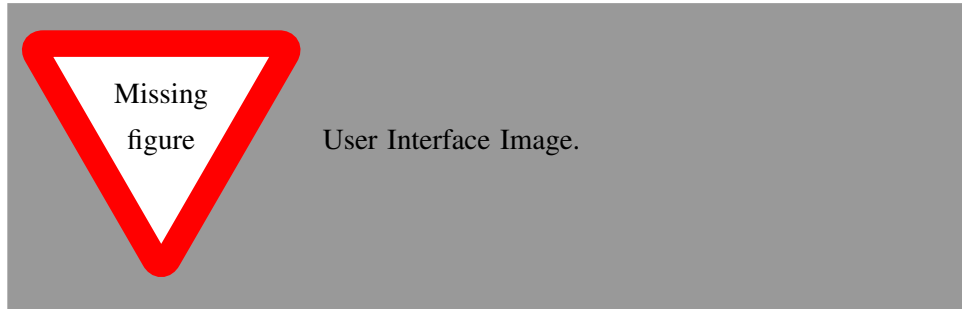


Figure 3: Algorithm interface class diagram.

bla

### 5.3 Workflow

### 5.4 User Interface



Why it  
looks  
like that.  
(Andere  
Soft-  
ware)

## **6 Result**

## **7 Conclusion**

## **8 Developer Guide**

### **8.1 Add Algorithm**

### **8.2 Edit UI**

## **Bibliography**

- [1] W. Burger and M. J. Burge, *Digital image processing: an algorithmic introduction using Java*. Springer, 2016.

## Glossary

**DWG** The native file format for AutoCAD data files.. 7

**DXF** The native file format for AutoCAD data files.. 7

**JVM** Java Virtual Machine. 7

**Kotlin** A statically-typed programming language that runs on the Java Virtual Machine.. 7

**OpenCV** A a library of programming functions mainly aimed at real-time computer vision.. 7

## List of Figures

1	AFPARS software architecture. . . . .	7
2	AFImage class diagram. . . . .	8
3	Algorithm interface class diagram. . . . .	8